

JDEM Database Prototyping

Igor Mandrichenko,
Vladimir Podstavkov

pgSphere

- Implements spherical geometry objects like point, circle, box, ellipse, polygon as “native” database data types
 - Uses GiST – Generalized Search Tree to speed up calculation of operations on these objects, e.g.:
 - Find all points within box
 - Find all points within circle
- But not:
- Find all points within distance from the point

Schema: Galaxy Catalog

Table "public.galcat"

Column	Type	Modifiers
galid	bigint	not null default nextval('galcat_galid_seq'::regclass)
coord	spoint	
row	double precision	
col	double precision	
mag	double precision	
radius	double precision	
dcoord	spoint	

Indexes:

"galcat_pkey" PRIMARY KEY, btree (galid)
"galcat_coord_idx" gist (coord)

Schema: Detections Catalog

```
Table "public.detcat"
Column | Type | Modifiers
-----+-----+-----
detid | bigint | not null default nextval('detcat_detid_seq'::regclass)
galid | bigint |
coord | spoint |
ftab | double precision[] |
fbin | bytea |
cgalid | bigint |

Indexes:
"detcat_pkey" PRIMARY KEY, btree (detid)
"detcat_cgalid_idx" btree (cgalid)
"detcat_coord_idx" gist (coord)
"detcat_galid_idx" btree (galid)

Foreign-key constraints:
"detcat_galid_fkey" FOREIGN KEY (galid) REFERENCES galcat(galid) ON UPDATE CASCADE
```

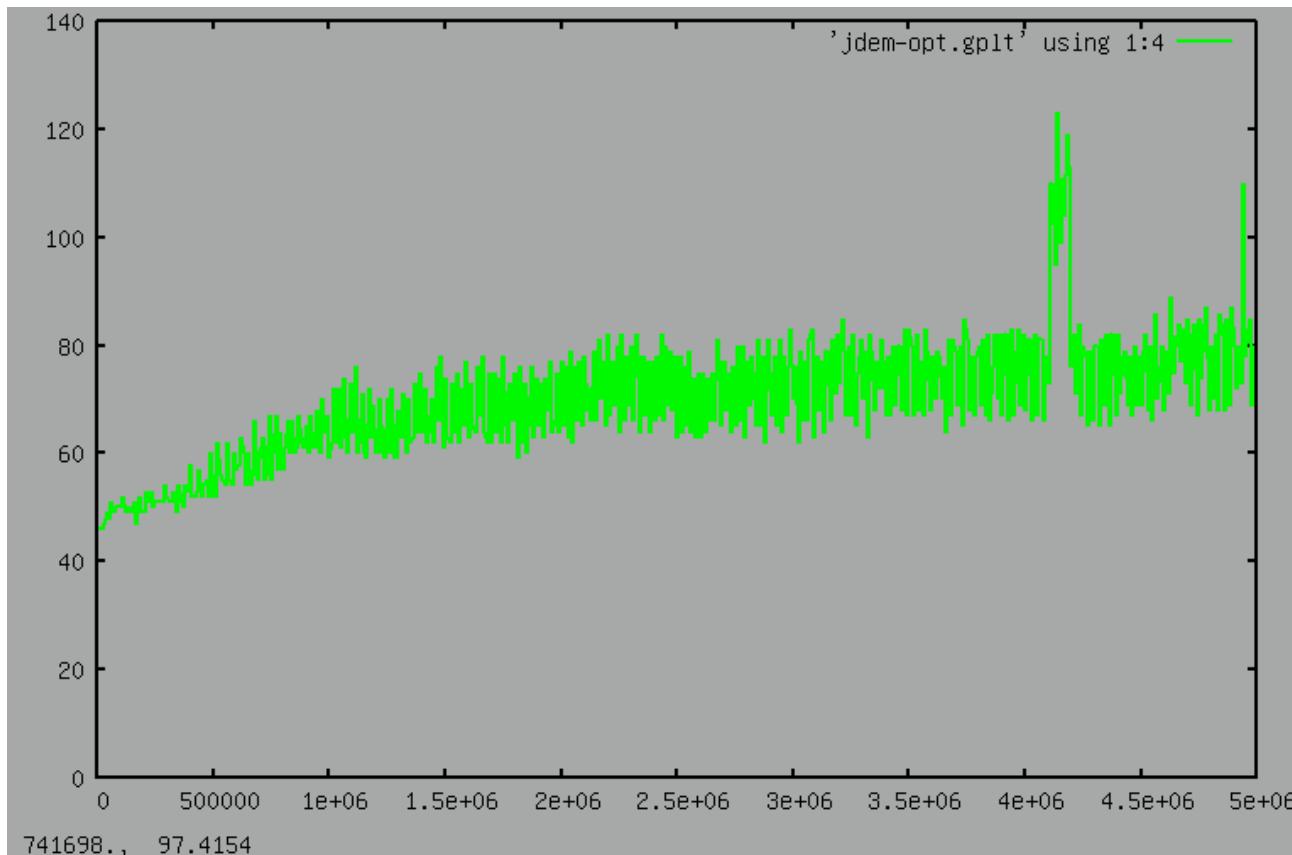
Schema: “Reconstructed” Galaxies

```
Table "public.calcat"
Column | Type | Modifiers
-----+-----+
galid | bigint | not null default nextval('calcat_galid_seq'::regclass)
coord | spoint |
Indexes:
"calcat_pkey" PRIMARY KEY, btree (galid)
"calcat_coord_idx" gist (coord)
```

Population

- We generated 1 day worth of JDEM Slitless Spectroscopy data:
 - 5 million galaxies (galcat)
 - 20 million detections – 4 for each galaxy (detcat)
 - Detection coordinates = galaxy coordinates + gaussian error with sigma=0.5 arc second
- Generation and population of the database took 10 hours of day time

Population timing



Time in seconds per 10,000 galaxies with 4 detections each

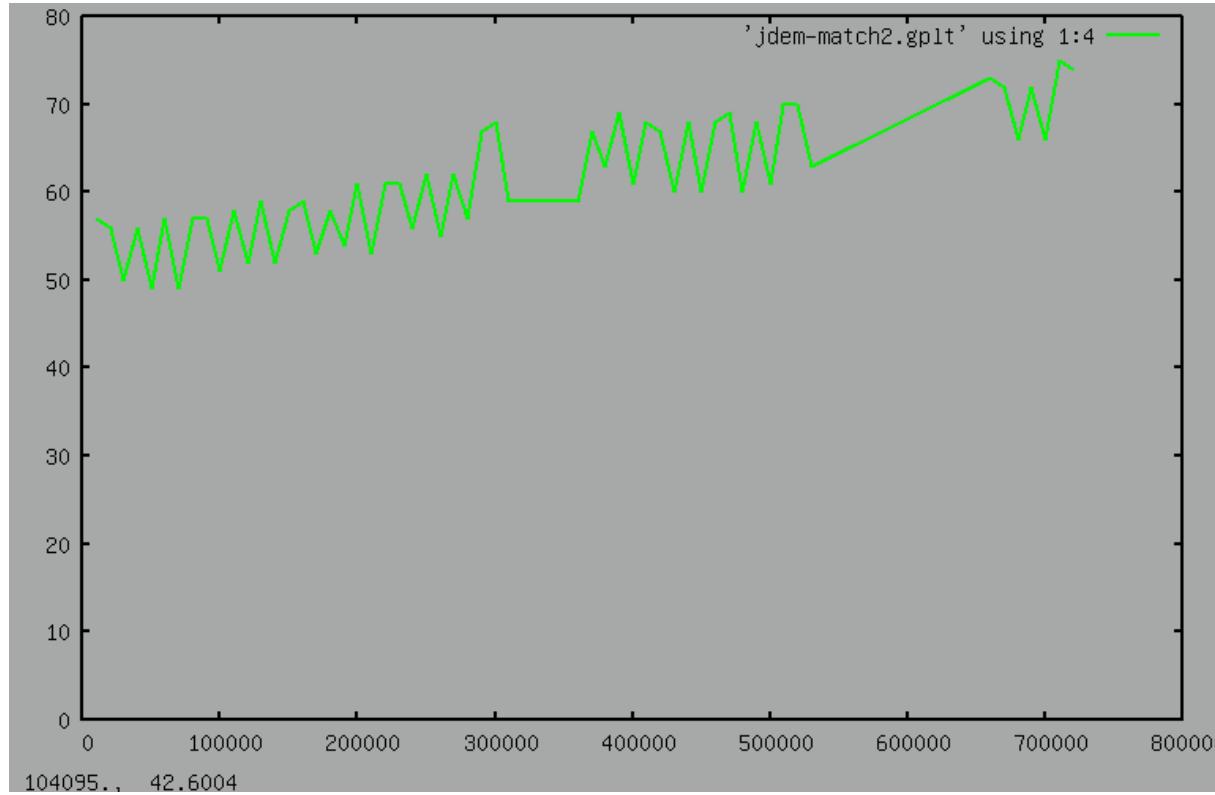
Galaxy “Reconstruction”

- To try using Postgres/pgSphere features, we implemented simple algorithm of galaxy location reconstruction from detections:
 - detcat -> calcat without using galcat

Galaxy reconstruction algorithm

1. get next record from detcat with cgalid=null
2. find a galaxy in calcat, nearest to the detection, within 2.5 arc seconds (5 sigma's),
3. if not found then
 1. store the detection as a new galaxy
 2. associate this detection with new galaxy (set cgalid) and go to (1)
4. if found, get all detections already associated with the galaxy (using cgalid) including this new one
5. find 4 of these detections closest to the galaxy coordinates calculated so far
6. if new detection is one of the 4 closest:
 1. set cgalid for 5-th point to null – disassociate old “bad” detection
 2. set cgalid for the new point to the galaxy id – associate new “better” detection
 3. recalculate galaxy coordinates as average coordinates of remaining 4 detections
7. else (new point is most distant of the 5):
 1. create new galaxy with coordinates equal to the coordinates of the new detection
 2. associate this new detection with new galaxy (set galid)
8. go to (1)

Reconstruction Timing



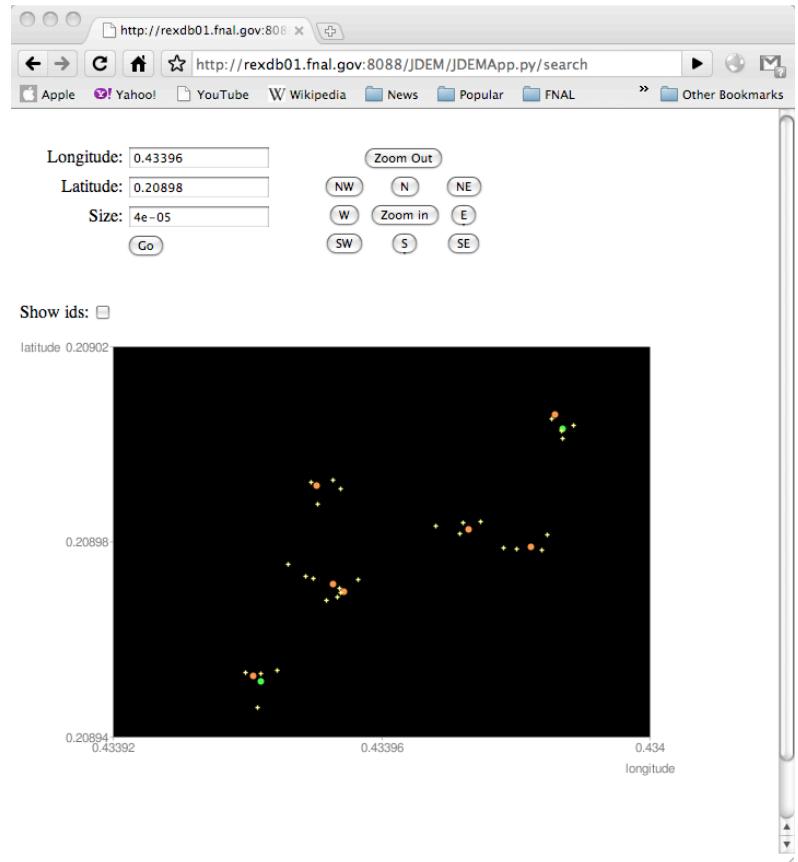
Time in seconds per 10,000 detections processed:

$\sim 100/10,000 = 100 \text{ Hz}$, but slows down with number of reconstructed galaxies

Current Database Server

- Old (>5 years) farm node with:
 - 4 cores
 - Intel(R) Xeon(TM) CPU 2.80GHz
 - 2 GB memory
 - 230 GB IDE disk
- Postgres 8.2.3
- Python 2.5
- Psycopg2

GUI to browse the database



<http://rexdb01.fnal.gov:8088/JDEM/JDEMApp.py/form>